AUSTRALIAN OS9 NEWSLETTER

sides 'No. of cylinders' (in decimal) :Interleave value: (in decimal) @FREE Syntax: Free [devname] Usage : Displays number of free sectors on a device @GFX Syntax: RUN GFX(<funct><args>) Usage : Graphics interface package for BASIC09 to do compatible VDG graphics commands @GFX2 Syntax: RUN GFX2([path]<funct><args>) Usage : Graphics interface package for BASIC09 to handle ax: none Usage : graphics/ window on-line help to lp topics @IDENT rmation from OS -s = use single lin xecution directory a device @INKE utine to input a abort to the pro nk to a memory ntents of text files @LOAD Syntax: Load <pathname> [...] Usage : Loads modules into memory @MAKDIR Syntax: Makdir <pathname> Usage : Creates a new directory file @MDIR Syntax: Mdir [e] Usage : Displays the present memory module directory Opts : e = print extended module directory @MERGE Syntax: Merge <path> ndard output @MFREE Synt RAM memory @MODPATCH a module in memory from warnings -c = compare modul name = link to module C o te V = verify module M = ma Montype [opt] Usage : Set mo monitor m = monochrome m sage : Creates and links an OS ROCS Syntax: Procs [e] Usage em Opts : e = display all proc e : Prints the current data directory path @PXD Syntax: Pxd Usage : Prints the current execution directory path @RENAME Syntax: Rename <filename> <new filename> Usage : Gives the file or directory a new name @RUNB Syntax: Runb <i-code module> Usage : BASIC09 run time package @SETIME Syntax: Setime [yy/mm TPR Syntax: ss to num @ reter @TMODE Syntax: Tmode [pathname] [params] Usage : Displays or changes the operating parameters of the terminal @TUNEPORT Tuneport </t1 or /p> [value] Adjust the baud value for the serial port @UNLINK Syntax: Unlink <modname> Usage : Unlinks module(s) from memory @WCREATE Syntax:

EDITOR	Gordon Bentzen	(07) 344-3881
SUB-EDITOR	Bob Devries	(07) 372-7816
TREASURER	Don Berrie	(07) 375-1284
LIBRARIAN	Jean-Pierre Jacquet	(07) 372-4675
	Fax Messages	(07) 372-8325
SUPPORT	Brisbane OS9 Users Group	

Addresses for Correspondence

Editorial Material:

Gordon Bentzen
8 Odin Street
SUNNYBANK Qld 4109

Subscriptions & Library Requests:

Jean-Pierre Jacquet
27 Hampton Street
DURACK Qld 4077

Volume 6 **April 1992** **Number 3**

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group
Volume 6 Number 3

EDITOR : Gordon Bentzen
SUBEDITOR : Bob Devries
TREASURER : Don Berrie
LIBRARIAN : Jean-Pierre Jacquet
SUPPORT : Brisbane OS9 Level 2 Users Group.

Hi there and welcome to another issue of your newsletter. It is good to see that we can still find enough new OS9 information to go into the newsletter, so that all our members can benefit from other OS9 users' knowledge.

Some of the information for this month's articles have come from the OS9 Community Network, which was mentioned last month. It has been a great source of all sorts of information, useful programming and hardware hints, and even new software details. It is pretty obvious, from reading the messages on the OCN echo, that although Tandy no longer supports the Colour Computer, its users will never give up, and are themselves providing new software for us to use.

I have been getting, on average, about 25 messages a day from the OCN echo, and many of thes pose questions that we have all faced at some time or other. They originate from all over the USA and Canada! I have 'chatted' with a user in Calgary, Alberta, and one in Honolulu, Hawaii. To send messages this way is much cheaper and quicker than by mail. I have posed questions, and received answers within 5 days. Even by airmail to USA a letter would take 10 to 14 days to be answered!

Amongst some of the information recently was an article detailing 'hidden' op-codes in Hitachi's 6309 processor chip, which is a direct 6809 replacement. That article, and a one about a new SCSI hard-disk driver, appear in this issue, as well as our old favourites, the C Tutorial, and the index of Rainbow OS9 articles.

Another of our articles is from a message on the OCN echo from a user whose place of work uses OS9 in its industrial format in dedicated process controllers.

Another useful aspect of the OS9 Community Network is, of course, the possibility to pose questions to other users on the network. We have printed some of these questions here so that hopefully you will be able to learn from them, too.

Through our association with the OCN we will have access to more PD software to add to our library. It will be very interesting to see what is becoming available in PD software in USA. One of the programmes which, I believe, will be coming over, is the RiBBS software package, which is an OS9 Bulletin Board System package. This package has full FIDONet compatibility with up/down loading of files, and messages. Any one interested in setting up an OS9 BBS in Australia?

I still have a feeling that some of you out there are not using your computers to their fullest capabilities. If you were, I feel sure that you would be asking more questions from us, but to date, not very many people have written letters asking questions! One of our new members, from Adelaide, did write, and I will share one of his questions with you all. He asked 'where can I get 96 TPI disks for my 80 track drives without paying $3.00 each for them?' My answer to him is: You don't need to use 96 TPI disks in the 720K 80 track drives. I have used standard 48 TPI disks in my 80 track drives, and I even buy the 'el cheapo' disks from a local computer shop, at about $5.00 per pack of 10 disks. I have not had any failures so far. Of course, for things like hard drive backups, it is advisable to use better quality disks, but still I only use 48 TPI disks, without problems.

His other question was about an article in 'The Rainbow' about installing an RS 232 port inside the CoCo 3. The problem he had was with the instructions for the modification of the T2 device descriptor. The instructions read:

```
modpatch /t2

c 10 68 30
v

ctrl-brk
```

Of course, the instructions should have read:

```
modpatch

1 t2
c 10 68 30
v

ctrl-brk
```

In my opinion it would seem that the author does not normally use modpatch for this operation, but perhaps uses DEd instead, but because everyone was supplied with modpatch, he decided to use that for patching. Pity he didn't check the instructions for modpatch first.

Cheers, Gordon.

OS9 and the Hitachi 6309 processor

The following message was received from an OS9 user in Calgary, Canada, and describes a report from an OS9 user in Japan, who modified his OS9 Level II machine (I don't think it was a CoCo) to enable it to use the extra functions of Hitachi's 6309 processor chip. The file '6309.txt' referred to in this message will be available from our PD library. It documents the extra programming instructions available in the 6309, but not how to change OS9 Level II to use them. Perhaps the necessary changes will be forthcoming from Kevin Darling or the other OS9 gurus in the US soon.

Conference: Os9
Message: 685 [PUBLIC]
From: Gerry McCleary
To: All
Subject: Hitachi CPU 6309 chip
Date: 03-12-92 17:49

Are you interested in the untold secrets of the HITACHI CPU 6309 CHIP!

Mr. Hirotsugu Kakugawa, of Computer Sys Lab., Faculty of Engineering, Hiroshima Univ., Japan. has done extensive work and has released his findings to the general public. This file is available on from my system or the KEYBOARD BBS [Calgary,AB (403) 246-6943] (1:134/67) in the OS-9 file [Area 20].

FILE NAME: MC6309.TXT (size 37399 bytes)

Here is an excert from the text file:

HITACHI says in the manual of 6309 that 6309 is compatible with 6809, but some OS-9 hackers found that it has secret features.

It has following features:

1. More registers (additional two 8 bit accumulators, 8 bit register, and a 16 bit register),
2. Two modes (6809 emulation mode and native mode),
3. Reduced execution cycles in native mode,
4. More instructions (16 bit x 16 bit multiplication, 32 bit / 16 bit division, inter-registers operation, block transfer, bit manipulating operation which is compatible with 6801 has, etc)
5. Error trap by illegal instruction, zero division.

I substituted 6309 for 6809 in my personal computer, and I changed OS9/6809 Level II such that the 6309 executes in native mode. I had to change the interrupt handling routine in the kernel. I implemented illegal instruction trap; I was really happy because most bugs are caught by trap handler.

Kevin Darling was very excited with the tests that he did after reading the message, so I'm sure that we can look forward to seeing more in this area.

ttyl

-=Gerry=-

--- Maximus 2.01wb
* Origin: The OS9 Keyboard [Calgary, AB 403-246-6943] (1:134/67)

oooooooooo000000000oooooooooo

Uses for OS9 in industry
from the OS9 Community Network on FIDONet

The following message was received from the OS9 echo on FIDONet. I thought it would be interesting to see what OS9 can be used for besides in our Colour Computers.

Conference: Os9
Message: 751 [PUBLIC]
From: Allen Morgan
To: Mike Warcholyk
Subject: Industrial OS9
Date: 03-13-92 10:23

Mike,

I work as an engineer in one of the largest Pulp & Paper plants in the world. We have installed several projects that use OS9-based control equipment and have developed some smaller applications ourselves. One system uses a Level-I 6809 with 256k CMOS static ramdisk. It takes wood-chip quality test data from a PC (one serial port), and takes other test data and about 250 limit-switch states from a programmable logic controller (2nd serial port), presents a compilation of all of this to an operator on a VT-100 terminal (3rd serial port), and sends all of this to a mini-computer(4th serial port).

It is actually a pretty complex application. The engineer spent about tree months on the code and debugging. It was all done in Basic09, written in modular fashion, and is very readable due to Basic09's long variable names. The physical environment makes the use of a hard drive or floppy a bad idea. The CMOS ram disk with lithium battery will withstand 140 degrees F, and lots of vibration. We use laptop computers to upload and download the code to the OS9 machine via one of the serial ports.

Another application uses several of the same OS9 computers to communicate between computerized scales, barcode readers, ultrasonic measurement systems, and a programmable logic controller. We also purchased a vibration analysis system for one of the paper machines which is OS9-68020. (actually 5 cpus) This system takes fast-Fourier transforms of the signals from accelerometers on 180 bearings on the machinery. The required bandwidth for the A/D converter is so high that the data transmission from the sensors to the CPU can't be done with Ethernet, it has to be analog. (512 FFT/sec). Results are kept on a 330 mbyte SCSI hard drive.

The software alarms if vibration "signatures" meet certain pattern requirements. Has 1024x768 Multisync color monitor, etc, etc. So, Mike, there is indeed a lot of OS9 used in industry. A lot of the applications are very complex. Being a good programmer is nice, but what is really in short supply is the engineer with an understanding of complex mechanical systems, the fundamentals of physics, and the practical side of wiring in the industrial environment.

Allen Morgan
--- RiBBS v2.02
 * Origin: RIBBS (1:105/641)

oooooooooo0000000000ooooooooo

A C Tutorial
Chapter 7 - Strings and Arrays

WHAT IS A STRING?

A string is a group of characters, usually letters of the alphabet. In order to format your printout in such a way that it looks nice, has meaningful titles and names, and is esthetically pleasing to you and the people using the output of your program, you need the ability to output text data. Actually you have already been using strings, because the second program in this tutorial, way back in Chapter 2, output a message that was handled internally as a string. A complete definition is a series of "char" type data terminated by a NULL character, which is a zero. When C is going to use a string of data in some way, either to compare it with another, output it, copy it to another string, or whatever, the functions are set up to do what they are called to do until a NULL, which is a zero, is detected.

WHAT IS AN ARRAY?

An array is a series of homogeneous pieces of data that are all identical in type, but the type can be quite complex as we will see when we get to the chapter of this tutorial discussing structures. A string is simply a special case of an array. The best way to see these principles is by use of an example, so load the program CHRSTRG.C and display it on your monitor. The first thing new is the line that defines a "char" type of data entity. The square brackets define an array subscript in C, and in the case of the data definition statement, the 5 in the brackets defines 5 data fields of type "char" all defined as the variable "name". In the C language, all subscripts start at 0 and increase by 1 each step up to the maximum which in this case is 4. We therefore have 5 "char" type variables named, "name[0]", "name[1]", "name[2]", "name[3]", and "name[4]". You must keep in mind that in C, the subscripts actually go from 0 to one less than the number defined in the definition statement.

HOW DO WE USE THE STRING?

The variable "name" is therefore a string which can hold up to 5 characters, but since we need room for the NULL character, there are actually only four useful characters. To load something useful into the string, we have 5 statements, each of which assigns one alphabetical character to one of the string characters. Finally, the last place in the string is filled with the numeral 0 as the end indicator and the string is complete. (A "define" would allow us to use "NULL" instead of a zero, and this would add greatly to the clarity of the program. It would be very obvious that this was a NULL and not simply a zero for some other purpose.) Now that we have the string, we will simply print it out with some other string data in the output statement. The %s is the output definition to output a string and the system will output characters starting with the first one in "name" until it comes to the NULL character, and it will quit. Notice that in the "printf" statement, only the variable name "name" needs to be given, with no subscript since we are interested in starting at the beginning. (There is actually another reason that only the variable name is given without brackets. The discussion of that topic will be given in the next chapter.)

OUTPUTTING PART OF A STRING

The next "printf" illustrates that we can output any single character of the string by using the "%c" and naming the particular character of "name" we want by including the subscript. The last "printf" illustrates how we can output part of the string by stating the starting point by using a subscript. The & specifies the address of "name[1]". We will study this in the next chapter but I thought you would benefit from a little glimpse ahead. This example may make you feel that strings are rather cumbersome to use since you have to set up each character one at a time. That is an incorrect conclusion because strings are very easy to use as we will see in the next example program. Compile and run this program.

SOME STRING SUBROUTINES

Load the example program STRINGS.C for an example of some ways to use strings. First we define four strings. Next we come to a new function that you will find very useful, the "strcpy" function, or string copy. It copies from one string to another until it comes to the NULL character. It is easy to remember which one gets copied to which if you think of them like an assignment statement. Thus if you were to say, for example, "x = 23;", the data is copied from the right entity to the left one. In the "strcpy" function, the data is also copied from the right entity to the left, so that after execution of the first statement, name1 will contain the string "Rosalinda", but without the double quotes, they are the compiler's way of knowing that you are defining a string. Likewise, "Zeke" is copied into "name2" by the second statement, then the "title" is copied. The title and both names are then printed out. Note that it is not necessary for the defined string to be exactly the same size as the string it will be called upon to store, only that it is at least as long as the string plus one more character for the NULL.

ALPHABETICAL SORTING OF STRINGS

The next function we will look at is the "strcmp" or the string compare function. It will return a 1 if the first string is larger than the second, zero if they are the same length and have the same characters, and -1 if the first string is smaller than the second. One of the strings, depending on the result of the compare is copied into the variable "mixed", and the largest name alphabetically is printed out. It should come as no surprise to you that "Zeke" wins because it is alphabetically larger, length doesn't matter, only the alphabet. It might be wise to mention that the result would also depend on whether the letters were upper or lower case. There are functions available with your C compiler to change the case of a string to all upper or all lower case if you desire. These will be used in an example program later in this tutorial.

COMBINING STRINGS

The last four statements have another new feature, the "strcat", or string concatenation function. This function simply adds the characters from one string onto the end of another string taking care to adjust the NULL so everything is still all right. In this case, "name1" is copied into "mixed", then two blanks are concatenated to "mixed", and finally "name2" is concatenated to the combination. The result is printed out with both names in the one variable "mixed". Strings are not difficult and are extremely useful. You should spend some time getting familiar with them before proceeding on to the next topic. Compile and run this program and observe the results for compliance with this definition.

AN ARRAY OF INTEGERS

Load the file INTARRAY.C and display it on your monitor for an example of an array of integers. Notice that the array is defined in much the same way we defined an array of char in order to do the string manipulations in the last section. We have 12 integer variables to work with not counting the one named "index". The names of the variables are "values[0]", "values[1]", ... , and "values[11]". Next we have a loop to assign nonsense, but well defined, data to each of the 12 variables, then print all 12 out. You should have no trouble following this program, but be sure you understand it. Compile and run it to see if it does what you expect it to do.

AN ARRAY OF FLOATING POINT DATA

Load and display the program named BIGARRAY.C for an example of a program with an array of "float" type data. This program has an extra feature to illustrate how strings can be initialized. The first line of the program illustrates to you how to initialize a string of characters. Notice that the square brackets are empty leaving it up to the compiler to count the characters and allocate enough space for our string. Another string is initialized in the body of the program but it must be declared "static" here. This prevents it from being allocated as an "automatic" variable and allows it to retain the string once the program is started. There is nothing else new here, the variables are assigned nonsense data and the results of all the nonsense are printed out along with a header. This program should also be easy for you to follow, so study it until you are sure of what it is doing before going on to the next topic.

GETTING DATA BACK FROM A FUNCTION

Back in chapter 5 when we studied functions, I hinted to you that there was a way to get data back from a function by using an array, and that is true. Load the

program PASSBACK.C for an example of doing that. In this program, we define an array of 20 variables named "matrix", then assign some nonsense data to the variables, and print out the first five. Then we call the function "dosome" taking along the entire array by putting the name of the array in the parentheses. The function "dosome" has a name in its parentheses also but it prefers to call the array "list". The function needs to be told that it is really getting an array passed to it and that the array is of type "int". The following line, prior to the bracket which starts the program, does that by defining "list" as an integer type variable and including the square brackets to indicate an array.

It is not necessary to tell the function how many elements are in the array, but you could if you so desired. Generally a function works with an array until some end-of-data marker is found, such as a NULL for a string, or some other previously defined data or pattern. Many times, another piece of data is passed to the function with a count of how many elements to work with. In our present illustration, we will use a fixed number of elements to keep it simple. So far nothing is different from the previous functions we have called except that we have passed more data points to the function this time than we ever have before, having passed 20 integer values. We print out the first 5 again to see if they did indeed get passed here. Then we add ten to each of the elements and print out the new values. Finally we return to the main program and print out the same 5 data points. We find that we have indeed modified the data in the function, and when we returned to the main program, we brought the changes back. Compile and run this program to verify this conclusion.

ARRAYS PASS DATA BOTH WAYS

We stated during our study of functions that when we passed data to a function, the system made a copy to use in the function which was thrown away when we returned. This is not the case with arrays. The actual array is passed to the function and the function can modify it any way it wishes to. The result of the modifications will be available back in the calling program. This may seem strange to you that arrays are handled differently from single point data, but they are. It really does make sense, but you will have to wait until we get to pointers to understand it.

A HINT AT A FUTURE LESSON

Another way of getting data back from a function to the calling program is by using pointers which we will cover in the next chapter. When we get there we will find that an array is in reality a pointer to a list of values. Don't let that worry you now, it will make sense when we get there. In the meantime concentrate on arrays and understand the basics of them because when we get to the study of structures we will be able to define some pretty elaborate arrays.

MULTIPLY DIMENSIONED ARRAYS

Load and display the file named MULTIARY.C for an example of a program with doubly dimensioned arrays. The variable "big" is an 8 by 8 array that contains 8 times 8 or 64 elements total. The first element is "big[0][0]", and the last is "big[7][7]". Another array named "huge" is also defined which is not square to illustrate that the array need not be square. Both are filled up with data, one representing a multiplication table and the other being formed into an addition table. To illustrate that individual elements can be modified at will, one of the elements of "big" is assigned the value from one of the elements of "huge" after being multiplied by 22. Next "big[2][2]" is assigned the arbitrary value of 5, and this value is used for the subscripts of the next assignment statement. The third assignment statement is in reality "big[5][5] = 177" because each of the subscripts contain the value 5. This is only done to illustrate that any valid expression can be used for a subscript. It must only meet two conditions, it must be an integer (although a "char" will work just as well), and it must be within the range of the subscript it is being used for. The entire matrix variable "big" is printed out in a square form so you can check the values to see if they did get set the way you expected them to.

PROGRAMMING EXERCISES

1. Write a program with three short strings, about 6 characters each, and use "strcpy" to copy "one", "two", and "three" into them. Concatenate the three strings into one string and print the result out 10 times.

2. Define two integer arrays, each 10 elements long, called "array1" and "array2". Using a loop, put some kind of nonsense data in each and add them term for term into another 10 element array named "arrays". Finally, print all results in a table with an index number.

   1. 2 + 10 = 12
   2. 4 + 20 = 24
   3. 6 + 30 = 36 etc.

   Hint; The print statement will be similar to; printf("%4d %4d + %4d = %4d\n",index,array1[index], array2[index],arrays[index]);

oooooooooo0000000000oooooooooo

An Index of Rainbow OS9 Articles
compiled by Bob Devries
January - December '89

January 1989 page 136
KISSable OS9 - BASIC09: a great language.
Dale L. Puckett

January 1989 page 152
Accessible Applications - OS9 memory explorations.
Richard A. White

February 1989 page 142
What day is it? - If OS9 date entries seem a little backward, this utility can help.
Richard Ries

February 1989 page 152
KISSable OS9 - Advances in OS9 technology.
Dale L. Puckett

March 1989 page 154
Accessable Applications - The importance of standard formats of directory use.
Richard A. White

March 1989 page 136
KISSable OS9 - Programs to tempt the DECB user.
Dale L. Puckett

April 1989 page 152
Accessible Applications - Data processing with Basic09.
Richard A. White

April 1989 page 148
KISSable OS9 - In quest of new technology.
Dale L. Puckett

April 1989 page 58
OS9 Resistance - Once you get to know it, OS9 is a great system.
Dennis Skala

May 1989 page 130
The Forgotten Chip - Get your modem to work with OS9 for under $20.
Carl Austin Bennett

May 1989 page 138
BASIC09 Programming Tool - Using syscall to enhance Basic09.
Philip Brown

May 1989 page 144
Chown - sharing those system files.
Evan Robinson

May 1989 page 146
Accessible Applications - More Basic09 programming.
Richard A. White

June 1989 page 136
PR.B09 - An OS9 printer utility.
Richard Ries

June 1989 page 150
KISSable OS9 - Building two handy tools.
Dale L. Puckett

July 1989 page 126
Syscall Sounds - A simpler way to call the SS.Tone system call.
Darrel Behrmann

July 1989 page 128
A CLS command for OS9 - Clear text to one background color.
Mark E. Sunderlin

July 1989 page 138
KISSable OS9 - Adding fireworks to Find.
Dale L. Puckett

August 1989 page 22
USTime & Stripbin - Two Basic09 utilities.
Jerry Yates

August 1989 page 66
OS9 survival training - The big scoop on OS9.
Jeffrey Parker

August 1989 page 122
KISSable OS9 - The big show in Chicago.
Dale L. Puckett

September 1989 page 112
Text File Compression - Getting the most out of disk storage.
Troy Brumley

September 1989 page 114
KISSable OS9 - MaxIc in Multi-Vue.
Dale L. Puckett

October 1989 page 72
Reach Out and Touch OS9 - Update your files.
Joseph Cheek

October 1989 page 82
What You Should Know About Your C Compiler - Getting

started with cgfx functions.
Numa David

October 1989 page 90
BASICally speaking - BASIC problems solved here.
Larry Boeldt

October 1989 page 110
KISSable OS9 - More on MaxIc.
Dale L. Puckett

November 1989 page 20
Mapping Your Finances - A matter of principle.
David Macias

November 1989 page 72
Observing the Social Graces - More than just "shaking hands".
Tim Koonce

November 1989 page 112
KISSable OS9 - Part III in the MaxIc series.
Dale L. Puckett

December 1989 page 106
Flipper09 - A familiar games of age in the OS9 arena.
Stephen J. Page

December 1989 page 121
Printing the Unprintable - A filter program for avoiding those "crazy" screens.
Richard Ries

December 1989 page 86
BreakPoint - An aid to checking file security and troubleshooting problems.
Greg Law

December 1989 page 110
KISSable OS9 - Comments from Kevin Darling.
Dale L. Puckett

oooooooooo0000000000ooooooooooo

CoCo-Link

CoCo-Link is an excellent magazine to help you with the RSDOS side of the Colour Computer. It is a bi-monthly magazine published by Mr. Robbie Dalzell. Send your subscriptions to:

CoCo-Link
31 Nedlands Crescent
Pt. Noarlunga Sth.
South Australia
Phone: (08) 3861647

oooooooooo0000000000ooooooooooo

Good Questions and Great Answers

Question 1 from Boisy G. Pitre
(Ex CoCo PUCC Internet List).

Basic09's stupidity

Yes! Basic09, as much as I have used it is STILL stupid!

Get this... I am trying to use a program with an assembly language subroutine called 'fpick'.

I have GFX2, SYSCALL, and INKEY merged in with my BASIC09 program file .. I typed in a sample program to call this 'fpick' sub....

It comes back with ERROR 043

So ok, I go in another window and type: load fpick

FPICK is loaded into memory. I go back to Basic09 and the program STILL gives me this stupid error. I'm frustrated beyond belief... I've had to fight this problem before, but never did get a clear explanation on this problem.

Is there a solution besides merging fpick into my BASIC09 program, which I DON't want to do?

Boisy

Answer from Kevin Darling
(Ex CoCo PUCC Internet List)

Which means either it couldn't find it (not true in this case, since it was loaded and we'll assume it had an a type of "6809 object subroutine"), OR...

It couldn't fit it into Basic09's own 64K map. It's really kind of easy:

8 . Each process gets up to eight 8K ramblocks

(8*8K = 64K).
-3 . Basic09 itself is 3 blocks long.
-1 . Basic09 usually starts using at least 1 block for data memory.
-1 . Gfx2/etc probably take up another block.
---
3 = block spaces left over, normally.

If "fpick" can't be mapped into the remaining 24K (3 blocks), then either:

a) it's bigger than 24K,

b) you started up basic09 with 32K of data memory and nothing can be mapped in at all, or

c) your program has already mapped in other subroutine or packed modules... or some combo.

There are ways around some of this; just let us know which it might be. BTW, with my PMap command (I'm sure it must be up here somewhere... guys?), [and available from our PD Library - Ed] then you can actually see how many blocks are left over in any process map.

best - kevin

And after reading what Kevin Darling had to say, Boisy replies:

Well here's the deal:

BASIC09 (w/ gfx2, syscall, and inkey merged) = 24K
the size of my program (including data) = 32K
the size of the fpick program (6809 subr.) = 7.1K

Thats ALMOST 64K... So anyways, I DID get it to work, but I have to PACK it first and run it w/ RUNB...

Boisy

Question 2 from Andrew Donaldson

What is the best way of closing a window opened by an application program? (This is on a CoCo, os9LII, from a C program using CGFX calls)

What I have been doing is:

When the user runs the program, store everything I can get about his current window, close the window, open one of the right type (or leave the current one if it was right), run the program, close the window, and restore the window as closely to the values saved above as possible.

The problem with this is that if someone uses a window which defaults to a non-full screen, and not the first window on the screen, type of window, the window dissapears, and is not selectable via the clear key.

This will crash the entire system if I do it twice!

Another way of doing this type of thing, is to create a new window, and copy the std in/out/error paths to it. The disadvantage of this is that the memory for the old screen is still used.

Thanks, Andrew.

Answer 2 from Don Berrie

To accomplish this, one simply needs to find all of the necessary information about the current window, and save it for later restoration. Principally, we need to store things like the current palette information, the current foreground, background and border colours, the window type and size, and the options section of the path descriptor. After the application has terminated, we simply reset all of the window infromation, and terminate.

Unfortunately, due to a bug in the standard cgfx.l library, (supplied with the Multivue system) the _gs_opt library call is not implemented, even though it is mentioned in the manual. Version 7 of Mike Sweet's replacement cgfx library, [available from our PD library .. Ed] does however have that call implemented. As well, it also has calls available to get all of the necessary information to reconstruct the original window.

I guess, in a way, this is really not the total answer to the question, as Andrew has stated that he wanted the application to totally redefine the original window in order to execute.

It has been my experience, that it is much better to open the next available window, using the "/w" descriptor, and setting things up as you really want them, rather than to modify the current window. However, as Andrew says, on limited systems, this may be inappropriate if there is insufficient memory to open the desired window. In my opinion, though, it will usually be better to trap for an out of memory error when opening the new window, to determine that case, rather than to redefine the current window, and it will also resolve problems associated with trying to reconstruct (one or a series of) overlays.

Question 3 from "coco cat"
(Ex CoCo PUCC Internet List)

Is there a such thing as a basic09 compiler?

The reason that I ask is that I cannot (and will not)

stand the editor that comes in basic09.

SO. I have other editors, written my own (not really that good, yet [still working on it]), and will probably get others that catch my eye. I don't want to edit the file, save it, then load it into basic09, and THEN pack it. I would like a compiler that takes the file and puts it in basic09 i-code.

Hmmm .... - coco cat.

Answer 3 from Philip Brown

(Ex CoCo PUCC Internet List)

Yeah... it's called basic09 :-)

You COULD write a shell-script that runs basic09, and tells it to load a certain file, then pack to another file .. but you'd miss all the debugging info.

So you have to keep a basic09 running (or fire it up every time you finish editing.. ugh)

Philip

ooooooooooOOOOOOOOOOoooooooooo

THE SCSI SYSTEM 1.0

Thanks to Leslie Donaldson, who posted this information onto the CoCo PUCC Internet List, after downloading it from DELPHI.

ANNOUNCING THE FASTEST, MOST VERSATILE COCO HARD DRIVE SYSTEM EVER!!!

If you have a Disto, Owl, Ken-Ton, RGB or LR-Tech SCSI interface, then you already have the hardware! Now all you need is this software!

- Support for 512 byte sectors. Get those extra megabytes from your Seagate N drives and others, and be able to use drives that don't support 256-byte sectors, such as those from Quantum.

Drive	Megabytes Right Now	Megabytes SCSI System	Extra
ST225N	20	21.3	1.3
ST125N	18	21.5	3.5
ST296N	80	85	5
ST138N	27	32	5
ST251N	36.4	43	6.6
ST157N	39.4	48.6	9.2
ST177N	51.6	61	9.4
ST277N	55	65	10
ST1096N	67.9	84	16.1

- Fastest read/write times of any CoCo hard drive system, including non-SCSI systems such as CoCo XT and The Eliminator. Who says SCSI is slow???

CoCo Megaread Times:

Other SCSI Drivers: 60+ seconds
Burke & Burke: 55 seconds
The Eliminator: 41 seconds
The SCSI System: 35 seconds

By the way, the "Megawrite" time for The SCSI System is just as fast.

- Despite the fast megaread, the multitasking is still super smooth. No interrupts are lost. Full F$Sleep support waiting for the drive to seek.

- Uses an intelligent write cache to handle the 256-to-512 conversion, thus allowing write times as fast as read times. Or you can select a read / modify / write mode of operation if you don't trust the cache.

- Other optional modpatches are provided to speed up RBFman even further and to smooth out the multitasking even more.

- Tells the Seagate drive to verify itself during format. No more mapped out sectors under OS/9. No more need for a long physical verify during format.

- All error mapping done by the drive.

- If a sector fails on a read or write, the driver can optionally send a command to tell the drive to reassign the block automatically, preventing the error from happening again (although, of course, the data in that sector is lost).

- Support for ALL CoCo SCSI interfaces, including Disto 4-in-1, Disto HDI, Ken-Ton, LR-Tech, RGB and Owl controllers. Supports Multipak and non-Multipak systems.

- A completely new SCSI utility that takes the guesswork out of formatting, modifying descriptors, etc. Also lets you send any command and data to the drive you want, and to view the results. No more flukey OS-9 format command.

No assembling device descriptor source files for your drive. Etc.

- Yet another new, improved Dmode command.

- Ability to send and receive any SCSI command and data whatsoever, using a new GetStat call. Potential applications allow for homemade diagnostic programs and support for SCSI tape drives and CD-ROMs.

- Robust handling of the SCSI protocol. Can recover from a hung-up drive, and takes into account the Seagate reset fluke. Also takes into account the mode select parameter saving "bug". Can return OS/9 errors or the raw SCSI error for debugging.

- And more. Stay tuned for details.

I started working on these drivers about a year and a half ago, and after all the bragging I did then, I finally got them done. Right now I'm still in the middle of my own gamma testing, but soon I will be looking for beta testers.

Although work on these drivers has been on-again, off-again, I've spent several hundred hours lately to crank out the package you see before you. Now I know why I put these drivers off so long ... it has taken one heckuvalot of work and experimentation to implement all the features and utilities you see here. Plus I blew a load to get a second H/D so I could develop these drivers and test them in a reasonable amount of time. This was part of the reason I kept putting them off ... I didn't want to have to work off of floppy and back up my hard drive everytime I wanted to reformat the disk and test the driver. That, combined with the fact I'm broke and can't afford to provide support for this package without some cash flow, means that I will be distributing The SCSI System for a price. A very low one, but enough to recover my costs. Keep watching Delphi [and here - Ed] for more details.

Matt Thompson

oooooooooo0000000000ooooooooooo

"Upgrade" Update

Much has been heard and written about the fabled OS9 Level II upgrade, which never actually made it (in total) to all of its users. So, as a service to the users, Leslie Donaldson has compiled this list of patches which are either already released into the public domain, or are in the pipeline (sic), to enable developers to have a "standardised" system at which to aim.

This file describes the upgrades or patches to be presented in the upgrade 2.5 packages. Since many patches (aka kludges) are going to be used and presented, I am going to use the name of the drivers affected by the patches . Following each driver is the patch type and possibly the author if the information is available.

Driver	Patch	Description and author
kernal	Yes	Filenames and memory size detection.
Rel	None	
Boot	Yes	Change the step rate of boot drive.
		mdopatch 00c0 03 00 017c 13 10
		K Darling K Meyers.
os9p1	Yes	Filenames (I have seen it for this and kernal)
os9p2	Yes	Fix The Sleep bug.
os9p3	New	Full length error messages.
os9p4	New	New command for register dump.
Ioman	Yes	Wrong register in inserting by priority into queues.
		09A6 10 12 09A7 A3 E1
		K Darling K Meyers.
Init	Yes	Virq Table size 000C 0F 0C
		default disk (personal - will not be changed)
		K Darling K Meyers.
CC3go	Yes	Allow use of shellplus parameter startup file
		Alternate startup files.
CC3io	Yes	Found in mv2pat fixes eating char register auto kill
		mouse button to unpause screen.

		Second patch to allow serial mouses(?)
		Cursor postion in compuserv B format.
Clock	Yes	Many patches including interrupts, and ticks
RBF	Yes	new RBF module allows for undelete.
CC3disk	Yes	Disto's no halt plus with sleep
		The IBM read disk Patch (pcdos)
		The Cache addition.
	Replacement	Uses sleep calls I belive to give better response.
CCHDscsi\		
CCHDsasi 〉	Replace	Announcement from M. Thompson will replace these three.
CCH3disk/		
SCF	Yes	Better line editing
ACIAPak	Yes	Buffer size / overun char buffer size changed
SACIA	Replacement	Better then Aciapak, more options.
bitbang	NEW	Allows for 1200 baud from bitbanger port.
Printer	None	
VRN	Replacement	Better memory control. replaces NIL,FTDD,VI
NIL	None	
GRFint	None	
WindInit	Yes	fixes a lot of little bugs Kent Meyers?? name mv2pat.ar
VDGinit	None	
PipeMan	None	
Piper	None	
GRFDRV	Yes	Faster Graphics K. Darling
		28 rows of text 9 bit characters and 25/24 lines of text
Shell	Replacement	Shell plus 2.1
		CHD and CD to allow open without write permission.
Gshell	Yes	Trashcan, new screen types, help option, new commands

This is a rough outline of the files and patches to be include in the package. I also want to include the ipatch util to make the entire package self contained.

I need suggestions hints or whatever from the readers of the list. Please send all comments to:

donaldlf@rosevc.rose-hulman.edu

Thank you for your time.

Note the patches I need so far are the upgrade version of cc3disk.dr. Please note this is a very fist draft so don't flame me yet. Please wait till the second draft to flame. :)

Leslie Donaldson

[If you care to respond, we will pass on all replies - Ed]

oooooooooo0000000000ooooooooooo